An event-driven method to simulate Filippov systems with accurate computing of sliding motions

PETRI T. PIIROINEN
Bristol Center for Applied Nonlinear Mathematics
Department of Engineering Mathematics
University of Bristol
Bristol BS8 1TR, United Kingdom

YURI A. KUZNETSOV
Mathematisch Instituut
Universiteit Utrecht
Budapestlaan 6, 3584 CD Utrecht, The Netherlands

June 7, 2005

Abstract

This paper describes how to use smooth solvers for simulation of a class of piecewise smooth dynamical systems, called Filippov systems, with discontinuous vector fields. In these systems constrained motion along a discontinuity surface (so-called sliding) is possible and require special treatment numerically. The introduced algorithms are based on an extension to Filippov's method to stabilize the sliding flow together with accurate detection of the entrance and exit of sliding regions. The methods are implemented in a general way in MATLAB and sufficient details are given to enable users to modify the code to run on arbitraray examples. Here, the method is used to compute the dynamics of three example systems, a dry-friction oscillator, a relay feedback system and a model of a oil well drill-string.

1 Introduction

Reliable numerical tools for simulation of mechanichal (e.g. gears and breaks) and electrical (e.g. relay systems and DC-DC converters) systems play an important part in the analysis and development of such systems. What characterizes these systems is that they are often modelled by sets of ordinary differential equations (ODEs) of varied complexity. There is a wide variety of numerical methods for solving ODEs and many of them are routinely used in established software, e.g. MATLAB [3]. However, most of these algorithms require that the ODEs are sufficiently smooth, while real world models such as these mentioned above typically include some kind of discontinuities. Systems with discontinuities are often referred to as *piecewise smooth* (PWS) systems, where the discontinuities could be either in the states or in the right hand sides (the vector field or its derivatives) of the ODEs (see e.g. [17]). These kind of systems require special numerical treatment during simulation which will be aparent below.

There are obvious differences in the treatment of systems with state jumps (e.g. due to impacts in mechanical systems) and discontinuous vector fields (e.g. due

to switches in electronics). In the present paper we will focus on the latter type, usually referred to as *Filippov systems* (see [17, 30]). The most important feature of Filippov systems is the possibility of motion constrained to some subset of the state space. Such constrained motion is often referred to as *sliding* [11, 13] (or *sticking* in the context of friction systems [19, 18]. Furthermore, from a dynamical systems point of view what characterizes general PWS systems is that they can not only undergo standard bifurcations (such as fold, flip and Hopf bifurcations) but also *nonsmooth transitions* (also referred to as *C-bifurcations*). In particular, Filippov systems can exhibit four generic types of codimension-one nonsmooth bifurcations of limit cycles, namely, the *adding-sliding*, *crossing-sliding*, *grazing-sliding*, and *switching-sliding* bifurcations [29, 13]. A particular feature of these nonsmooth transitions is the possibility of sudden onset of chaos or jumps to nonlocal attractors, which cannot be seen in smooth systems (see e.g. [21, 28]).

A first step towards understanding dynamics at these nonsmooth transitions is often to perform direct numerical simulation (DNS), where it is of great importance that the time and location of any nonsmooth events are resolved as accurately as possible, e.g. in a nonsmooth system solver [20]. This idea can be compared with another idea for simulating nonsmooth systems, which is to recast the nonsmoothness in terms of a complimentarity system formulation [4]. Then one can use time stepping methods accompanied with linear complemetarity problem (LCP) solvers to similate the systems without the need for accurate event detection. That is, the solver can only note that one or more events have occured during a time step without finding the actual event time and location (see further [32, 26]). Such methods have proven to be effective in simulating mechanical systems with a large number of constraints. However, they suffer from from the disadvantage that they are typically only low-order algorithms and nonsmooth events can be lost. The focus of this work is to accurately detect nonsmooth transitions and therefore we require high order algorithms to solve the smooth vector fields, a way of stabilize the sliding flow and accurate event detection algorithm. Such a simulator can then be used to compute Poincaré maps and to continue limit-cycles and their sliding bifurcations under parameter changes in general Filippov systems, as fixed points of these maps. Up to now, most analysis of Filippov systems has been (semi-)analytic, and usually limited to small systems [13]. However, a numerical tool that has succesfully been implemented to analyse Filippov systems is SlideCont [9]. It is based on the widely used numerical package auto [14] that can continue solutions to nonlinear boundary-value problems via orthogonal collocation. To some extent, SlideCont has the ability to continue equilibria, limit cycles, and their sliding bifurcations, but to date it still lacks the capability to perform DNS of Filippov systems and automatically switch between sliding and nonsliding motions. In this paper an algorithm for simulation of Filippov systems will be presented that precisely fills this gap, i.e. it solves for the sliding flow directly and automatically switches between free and constrained motion. Since only smooth systems are to be solved numerically, well known methods together with appropriate error estimates can be applied between the switches.

The rest of the paper is organized as follows. In sec. 2 we introduce and define Filippov systems and explain what characterises them. A description of the numerical algorithm for simulating these Filippov systems is presented in sec. 3. Three examples, a dry-friction oscillator, a relay feedback system and a drill-string model, are presented in sec. 4, along with instructions for users who wants to use

the downloadable programs. The actual MATLAB files used for simulation of Filippov systems are presented and explained in Appendix A, including instructions for potential users. Finally, in sec. 5 we conclude this paper and discuss future work on methods to locate and continue limit cycles and sliding bifurcations of Filippov systems.

2 Filippov systems

As mentioned in sec. 1 we will consider dynamical systems with discontinuous vector fields, so called Filippov systems. What characterizes such a system is the division of the state space into disjoint subregions, such that in each such region the defining vector field is smooth. The boundaries between the different regions will be referred to as *discontinuity surfaces*. In this section only a brief introduction to Filippov systems will be given, and for a more thorough exposition of this topic see [17, 30, 13, 29].

A general dynamical system can be written as

$$\dot{x} = f(x), \quad x \in \mathbb{R}^n, \tag{1}$$

where the vector field $f(x)$ can be either smooth or piecewise smooth. To begin with, let us assume that the state space consists of only two regions, S_i and S_j, separated by a discontinuity surface Σ_{ij}, which is defined by a smooth scalar function $h_{ij}(x)$ such that

$$\Sigma_{ij} = \{x \in \mathbb{R}^n \mid h_{ij}(x) = 0\}, \tag{2}$$

and where

$$S_i = \{x \in \mathbb{R}^n \mid h_{ij}(x) > 0\} \text{ and } S_j = \{x \in \mathbb{R}^n \mid h_{ij}(x) < 0\}. \tag{3}$$

Hence, (1) can be rewritten as

$$\dot{x} = \begin{cases} F_i(x), & x \in S_i, \\ F_j(x), & x \in S_j, \end{cases} \tag{4}$$

where F_i and F_j are sufficiently smooth. In the rest of this paper it is also assumed that F_i and F_j are defined in the whole state space, even if they are only used in their respective regions.

If the vector fields F_i and F_j are locally both pointing away from or towards the discontinuity surface Σ_{ij} the dynamics is assumed to be locally constrained to the surface, as depicted in the left part of Fig. 1, and the motion is said to be sliding. As mention in sec. 1 mechanical systems with friction one often refers to sliding as sticking. The open subset $\hat{\Sigma}_{ij}$ of the surface Σ_{ij} where the vector fields are both pointing towards or away from Σ_{ij} is often referred to as the *sliding surface*. In Fig. 1 the sliding surface $\hat{\Sigma}_{ij} \subset \Sigma_{ij}$ is a line segment between two points, $\hat{\Sigma}_{ij}^-$ and $\hat{\Sigma}_{ij}^+$. If it holds that

$$\mathcal{L}_{F_i - F_j}(h_{ij})(x) < 0, \quad x \in \hat{\Sigma}_{ij}, \tag{5}$$

then $\hat{\Sigma}_{ij}$ is stable, while if

$$\mathcal{L}_{F_i - F_j}(h_{ij})(x) > 0, \quad x \in \hat{\Sigma}_{ij}, \tag{6}$$

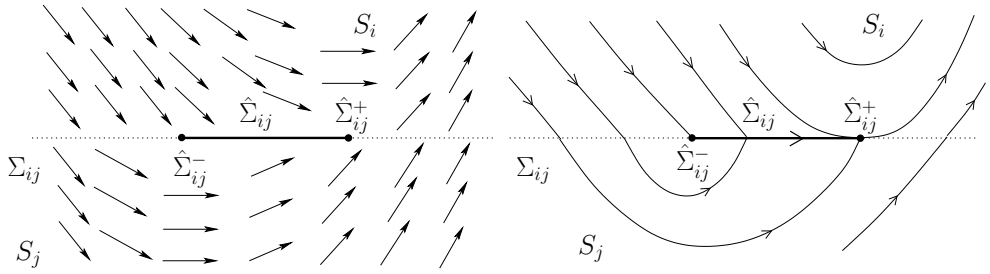

Figure 1: Vector fields F_i and F_j near a stable sliding region $\hat{\Sigma}_{ij}$ (left) and the corresponding orbits (right).

the sliding surface is unstable. Here $\mathcal{L}_F(h)(x)$ is given by

$$\mathcal{L}_F(h)(x) := \frac{dh}{dt}(x) = \frac{dh}{dx}\frac{dx}{dt}(x) = \left\langle \frac{dh}{dx}(x), F(x) \right\rangle \tag{7}$$

and is sometimes referred to as the *Lie derivative* of $h(x)$ along $F(x)$ (see e.g. [1]).

Using *Utkin's eqvivalent control* [37] the dynamical system (4) can be extended to include the vector field on the sliding surface such that

$$\dot{x} = F_{ij}(x), \quad x \in \hat{\Sigma}_{ij}, \tag{8}$$

where

$$F_{ij}(x) = \frac{F_i(x) + F_j(x)}{2} + \frac{F_j(x) - F_i(x)}{2}\mu_{ij}(x), \tag{9}$$

and $-1 \leq \mu_{ij}(x) \leq 1$. Since the motion is constrained to $\hat{\Sigma}_{ij}$, F_{ij} must be tangent to $\hat{\Sigma}_{ij}$, i.e. $\mathcal{L}_{F_{ij}}(h_{ij})(x) = 0$, which yields

$$\mu_{ij}(x) = -\frac{\mathcal{L}_{F_i+F_j}(h_{ij})(x)}{\mathcal{L}_{F_j-F_i}(h_{ij})(x)}. \tag{10}$$

It is notable that $F_{ij} = F_i$ and $F_{ij} = F_j$ when $\mu_{ij}(x) = -1$ and $\mu_{ij}(x) = 1$, respectively, and also $\mathcal{L}_{F_j-F_i}(h_{ij})(x) \neq 0$ for $x \in \hat{\Sigma}_{ij}$ since the vector fields always point toward or away from the discontinuty surface. The surfaces defining the borders of the sliding surface are thus given by

$$\hat{\Sigma}^-_{ij} = \left\{ x \in \hat{\Sigma}_{ij} \mid \mu_{ij}(x) = -1 \right\} \text{ and } \hat{\Sigma}^+_{ij} = \left\{ x \in \hat{\Sigma}_{ij} \mid \mu_{ij}(x) = 1 \right\}. \tag{11}$$

These borders are composed of so called *tangent points* [29] and will be referred to as *tangent surfaces* (cf. the points $\hat{\Sigma}^-_{ij}$ and $\hat{\Sigma}^-_{ij}$ in Fig. 1).

We next define solutions of the Filippov system (4) by concatenating standard solutions in S_i and S_j and sliding solutions in $\hat{\Sigma}_{ij}$ (see the right part of Fig. 1). To assure the uniqueness of such a solution, it is sufficient to assume that it does not visit points of $\hat{\Sigma}_{ij}$, where *both* vectors F_i and F_j are tangent to Σ_{ij}. In what follows, we call such solutions and their corresponding orbits *generic*.

Finally, let us discuss briefly a relationship between the above desribed construction and the commonly used *Filippov's convex method* [17] (see further sec. 3.6). The original Filippov approach consists of replacing (4) by the differential inclusion

$$\dot{x} \in \begin{cases} \{F_i(x)\}, & x \in S_i, \\ \overline{\text{co}}(F_i, F_j), & x \in \Sigma_{ij}, \\ \{F_j(x)\}, & x \in S_j, \end{cases} \tag{12}$$

where $\overline{\text{co}}(F_i, F_j)$ is the minimal closed convex set containing F_i and F_j, i.e.

$$\overline{\text{co}}(F_i, F_j) = \{f \in \mathbb{R}^n : f = \lambda F_i + (1-\lambda)F_j, \lambda \in [0,1]\}.$$

A solution to this differential inclusion is an absolutely continuous function $x(t)$ that satisfies (12) for almost all t from its domain of definition. Under assumed smoothness of F_{ij} and h_{ij}, the famous THEOREM 2 by [17](Chapter 2, pp. 110-111) implies forward uniqueness of those solutions to (12), which do not visit points of Σ_{ij}, where both vectors F_i and F_j are tangent to Σ_{ij}. Moreover, this unique solution coincides with the one constructed above. It should be noted that in Filippov's approach solutions starting at unstable sliding surfaces are not unique, while they generically have this property in our formulation, where they are constrained to $\hat{\Sigma}_{ij}$. This differnce is not important in most applications, since solutions starting off the unstable sliding surface can never reach it.

3 Simulation of Filippov systems

Under some circumstances (e.g. if small linear systems are considered) it is certainly possible to find explicit expressions for the solutions of the ODE that describes sliding if the vector fields in the non-sliding regions are given. However, the idea here is to present a numerical algorithm where the user only provides the different vector fields and information about the discontinuity surfaces and then the vector fields for the sliding regions are automatically computed by a routine which uses eqs. (2), (8)-(10). The method that has been chosen here to simulate Filippov systems is similar to the hybrid system approach, where integrations of smooth ODEs are mixed with discrete maps and vector field switches. In practice, it means that an initial value problem is solved for one of the possible smooth dynamical systems given in eq. (1) until the orbit reaches one of the predefined surfaces. At such a point the vector field is possibly switched, depending on the state at that time instance (see futher sec. 3.2).

It will be assumed that a suitable time stepping method is chosen to integrate smooth ODE systems to a desired tolerance (within certain limitations). There are numerous methods to solve different types of ODEs (see, e.g. [22, 23, 35]) and based on them numerical solvers (e.g. [25, 5]) so the first task is, indeed, to choose one that is suitable for the current system. One of the most frequently used time-stepping solvers for dissipative systems is the fourth-order Runge-Kutta [6] that works very well for nonstiff problems, but for more stiff systems a special solver is required, based on, e.g., a BDF [5] or Radau [8, 24] method. For linear systems there also exist special methods [15, 33, 7], which are usually faster than general solvers for nonlinear systems. Because of our choice of strategy to integrate Filippov systems, it is very important to have a reliable ODE solver that is accompanied by an accurate routine to locate discontinuity and tangent surface crossings (see sec. 2). In what follows a surface crossing will be called an *event* and a scalar function defining an *event surface* is referred to as the *event function*. The existence of event detection routines will be here assumed. For instance in MATLAB event detection routines are built-in and can easily be used toghether with the likewise built-in ODE solvers to integrate orbits and to locate events along them as precisly as the accuracy of MATLAB allows (for more details of the MATLAB ODE routines, see [36, 3]). However, standard methods, e.g. secant type methods, can easily be implemented and have proven to be fast and reliable. The type of events to be detected also play

an important role in how to numerically deal with them and how sensitive the event detection needs to be.

In the rest of this section we will focus on special measures that have to be taken in order to have a Filippov solver that automatically switches between different vector fields, and that is relatively robust.

3.1 Regularizing the sliding vector field

Since a numerically constructed sliding solution might not follow the discontinuity surface exactly, we have to consider the vector field F_{ij} not only for $x \in \hat{\Sigma}_{ij}$ but in a neighbourhood of $\hat{\Sigma}_{ij}$. Under the above introduced generisity conditions, equations (9) and (10) can be used to define the vector field F_{ij} in such a neighbourhood, if we formally allow $|\mu_{ij}(x)|$ to be greater than 1. Note that orbits of the extended vector field F_{ij} pass through the boundary of the siding surface $\hat{\Sigma}_{ij}$, i.e. the tangent surfaces $\hat{\Sigma}^{\pm}_{ij}$. This property, that is illustrated in Fig. 2 (left), is essential for the event detection and location descibed in the next section. However, this extended

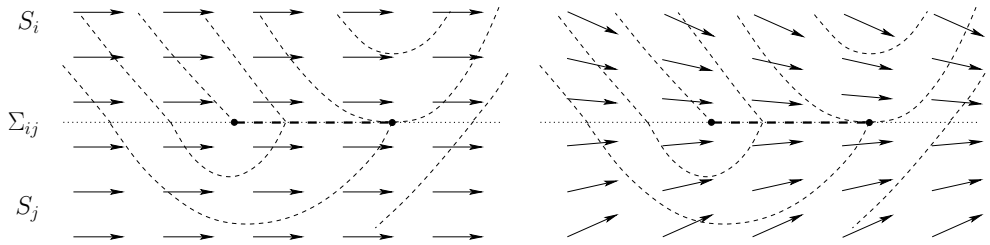

Figure 2: The sliding vector fields F_{ij} (left) and $\hat{F}_{ij}$ (right) in a neighbourhood of the discontinuity surface Σ_{ij}.

vector field has a family of *invariant surfaces* $h_{ij} = \text{const}$, one of which is Σ_{ij} given by $h_{ij} = 0$. Therefore, for a numerical solution of the sliding equation, there is the (unwanted) possibility of drifting away from the discontinuity surface, to one of the other invariant surfaces. This occurs due to accumulation of numerical errors combined with the neutral stability of the surface $\hat{\Sigma}_{ij}$ for $\dot{x} = F_{ij}(x)$ (see the left panel of Fig. 2). One way to avoid such a drift is to make the sliding surface $\hat{\Sigma}_{ij}$ attracting as long as the motion governed by the sliding vector field (9) is constrained to it. This can be done by introducing a new sliding vector field $\hat{F}_{ij}$ by adding a *small* term to the original vector field F_{ij} that makes the sliding surface *locally attractive* (see the right panel of Fig. 2). For example, this vector field can be defined as

$$\hat{F}_{ij}(x) = F_{ij}(x) - Ch_{ij}(x)\left(\frac{dh_{ij}}{dx}(x)\right)^T, \tag{13}$$

where C is a positive constant. It is clear that $\hat{\Sigma}_{ij}$ is locally attracting since the new term is orthogonal to $\hat{\Sigma}_{ij}$ and points towards the surface. Furthermore, for $x \in \hat{\Sigma}_{ij}$, we have

$$Ch_{ij}(x)\left(\frac{dh_{ij}}{dx}(x)\right)^T = 0,$$

which is exactly what we want since the extra term does not interfere with the sliding vector field as long as the solution stays on Σ_{ij}. It is always possible to

choose $C > 0$ such that the motion in the normal direction will be faster than sliding along the discontinuity surface. However, one should avoid choosing C too big because in this case the ODE system $\dot{x} = \hat{F}_{ij}(x)$ becomes stiff. Similar techniques to make constraint surfaces attractive are used in numerical integration of differential-algebraic equations (DAEs) (see e.g. [2]).

When $|\mu_{ij}(x)| > 1$ for $x \in \Sigma_{ij}$, then the motion will not be locally constrained to Σ_{ij}, and we note that $F_{ij} = \hat{F}_{ij} = F_i$ if $\mu_{ij} = -1$ and $F_{ij} = \hat{F}_{ij} = F_j$ if $\mu_{ij} = -1$. Therefore, to compute a solution passing through a point where $|\mu_{ij}(x)| > 1$, we must switch to F_i if $\mu_{ij}(x) < -1$ and F_j if $\mu_{ij}(x) > 1$.

3.2 Event location

When simulating Filippov systems using an event driven scheme it is important to locate events, e.g. discontinuity surface or a sliding boundary crossing, as accurately as possible (within a given tolerance). Therefore, to make an automatic algorithm robust, specific events need to be prediefined as region-dependant. If it is assumed that a system has a total of m possible events in each region then an *event list* $e(x,t)$ can be defined as

$$e(x,t) = (e_1(x,t), \ldots, e_m(x,t)), \tag{14}$$

where each element $e_k(x,t) \in \mathbb{R}$ is an event function that defines an event surface that can be reached by the state vector or the time.

To make things more clear, we will assume that there is locally only one discontinuity surface (Σ_{ij}) present. However, in sec. 4.3 a system with two discontinuity surfaces will be presented and numerically examined (see further sec. 3.6). As mentioned in sec. 2 the state space for a Filippov system with one discontinuity surface is divided into three interesting regions, namely, S_i, S_j, and Σ_{ij}. Further, as seen in Fig. 3, the state space can also be divided into two regions, $\hat{M}_{ij}$ and M_{ij}, by the two *extended tangent surfaces* Σ_{ij}^- and Σ_{ij}^+, which are defined by

$$\Sigma_{ij}^- = \{x \in \mathbb{R}^n \mid \mu_{ij}(x) = -1\} = \{x \in \mathbb{R}^n \mid \mathcal{L}_{F_i}(h_{ij}) = 0\}, \tag{15}$$

$$\Sigma_{ij}^+ = \{x \in \mathbb{R}^n \mid \mu_{ij}(x) = 1\} = \left\{x \in \mathbb{R}^n \mid \mathcal{L}_{F_j}(h_{ij}) = 0\right\}, \tag{16}$$

where μ_{ij} is defined as in eq. (10). For later reference, by using (15) and (16) the

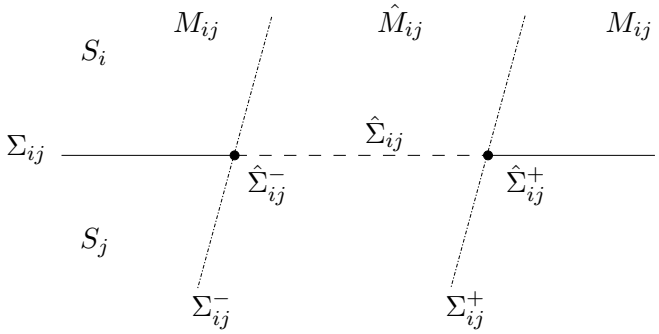

Figure 3: The different regions that the state space is divided into in a neighbourhood of the sliding surface $\hat{\Sigma}_{ij}$.

two regions $\hat{M}_{ij}$ and M_{ij} are defined as

$$\hat{M}_{ij} = \{x \in \mathbb{R}^n \mid |\mu_{ij}(x)| < 1\} \text{ and } M_{ij} = \{x \in \mathbb{R}^n \mid |\mu_{ij}(x)| \geq 1\}. \tag{17}$$

Notice the similarity with the tangent surfaces defined in eq. (11). Also, since it is assumed that F_i and F_j are defined in the whole state space, μ_{ij} is defined everywhere except for points where both F_i and F_j are tangent to the surface $h_{ij}(x) = \texttt{const}$. Recall that such points have been excluded from generic orbits in sec. 2.

This division into four disjoint subregions makes it relatively straightforward to implement in a numerical algorithm and reduces the number of checks that have to be made every time the discontinuity surface Σ_{ij} is crossed. Also, since we are always looking for the surfaces given by $\mathcal{L}_{F_i}(h_{ij})(x) = 0$ and $\mathcal{L}_{F_j}(h_{ij})(x) = 0$ it makes the algorithm more robust for the location of events for orbits that hit the discontinuity surface almost tangentially.

3.2.1 Event functions

For each region we will now introduce a number of event functions that will be used for the location of possible surface crossings, at which the state leaves one region and continues into another.

Event functions in S_i and S_j

If $x \in S_i \vee S_j$ there are three surfaces to look for, namely the discontinuity surface Σ_{ij}, and the extended sliding boundaries Σ_{ij}^- and Σ_{ij}^+ (see Fig. 3). Therefore the natural choices for event functions are $h_{ij}(x) = 0$, $\mathcal{L}_{F_i}(h_{ij})(x) = 0$ and $\mathcal{L}_{F_j}(h_{ij})(x) = 0$. However, depending on in which region the state x is in before the surface crossing it is enough to look for surface crossings of Σ_{ij} from one direction. For instance if $x \in S_i$ ($x \in S_j$) we know that $h_{ij}(x) > 0$ ($h_{ij}(x) < 0$) and thus we need only look for crossings of Σ_{ij} as h_{ij} changes from positive to negative (negative to positive). Similarily, by keeping track of the sign of the $\mathcal{L}_{F_i}(h_{ij})(x)$ and $\mathcal{L}_{F_j}(h_{ij})(x)$ one can determine from what direction to look for the surfaces Σ_{ij}^- and Σ_{ij}^+, respectively.

Event functions on the sliding surface $\hat{\Sigma}_{ij}$

If $x \in \hat{\Sigma}_{ij}$ there are two surfaces to look for, namely Σ_{ij}^- and Σ_{ij}^+ (see Fig. 3), and the corresponding event functions are $\mathcal{L}_{F_i}(h_{ij})(x) = 0$ and $\mathcal{L}_{F_j}(h_{ij})(x) = 0$, respectively. The search directions of these surfaces are found in the same way as the previous case.

Event functions directions

Following the discussion above each of our original event functions e_k can be seen as one of two different kinds, namely $e_k^+(x,t)$ and $e_k^-(x,t)$, where the former means that a surface crossings is only detected when e_k is increasing and decreasing, respectively.

3.2.2 Event variables

In order for the solver to know which vector field to use and which events to look for, a number of event variables will be introduced. Thus, to keep track of which of the regions S_i, S_j, and $\hat{\Sigma}_{ij}$ the state variable $x(t)$ is in we introduce the event variables s_1, s_2, and s_3, respectively. Further, to keep track of if the state variable is in M_{ij} or $\hat{M}_{ij}$ we introduce the event variables s_4 and s_5, respectively. Letting

$s = (s_1, s_2, s_3, s_4, s_5)^T$ und using the informations from sec. 3.2.1 we can give the event variables the following values

$$s_1(x) = \begin{cases} 1, & x \in S_i, \\ -1, & x \notin S_i, \end{cases} \quad s_2(x) = \begin{cases} 1, & x \in S_j, \\ -1, & x \notin S_j, \end{cases} \tag{18}$$

$$s_3(x) = \begin{cases} 1, & x \in \hat{\Sigma}_{ij}, \\ -1, & x \notin \hat{\Sigma}_{ij}, \end{cases} \tag{19}$$

$$s_4(x) = \begin{cases} 1, & x \in M_{ij}, \\ -1, & x \notin M_{ij}, \end{cases} \quad s_5(x) = \begin{cases} 1, & x \in \hat{M}_{ij}, \\ -1, & x \notin \hat{M}_{ij}, \end{cases} \tag{20}$$

Depending on in which region $x(t)$ is in and to which region it will continue the event parameters are changed at that event accordingly. For instance, assume $x \in S_i \cup \hat{M}_{ij}$ before it crosses the discontinuity surface Σ_{ij} then we will have

$$s_{ij} = (1, -1, -1, -1, 1)$$

before the crossing and

$$s_{ij} = (-1, -1, 1, -1, 1)$$

after. This means that the orbit is sliding along $\hat{\Sigma}_{ij}$ after the surface crossing and this information is passed to the solver so that the correct vector field is used in the solving process.

For any event variable change it is of course possible to predefine an *event matrix* such that an event variable vector (just before an event) is multiplied by a event matrix to get the new event variables (just after an event). However, this has not been used in the present version of the algorithm.

3.3 The dynamical system

Now we are ready to write the full dynamical system with one discontinuity surface as

$$\dot{x} = \begin{cases} F_i(x), & x \in S_i, \\ \hat{F}_{ij}(x), & x \in \hat{\Sigma}_{ij}, \\ F_j(x), & x \in S_j, \end{cases} \tag{21}$$

where the sliding vector field is given by

$$\hat{F}_{ij}(x) = \frac{F_i(x) + F_j(x)}{2} + \frac{F_j(x) - F_i(x)}{2}\mu_{ij}(x) - Ch_{ij}(x)\frac{d}{dx}h_{ij}^T(x), \tag{22}$$

and $\mu_{ij}(x)$ is defined by (10). Together with the laws (18)-(20) for the event variables if we let the event functions be defined by

$$e_1(x,t) := h_{ij}(x), \quad e_2(x,t) := \mathcal{L}_{F_i}(h_{ij})(x), \quad e_3(x,t) := \mathcal{L}_{F_j}(h_{ij})(x),$$

this constitutes equations to simulate a Filippov system using the event-driven approach.

In the table below we have listed what surface to look for in each region, the list of event variables in each region and also from which direction a zero crossing of an event function is to be looked for. A dash in the table means that there is no need to look for this zero crossing since this event cannot happen before another event has happened before.

$x \in$	$s_{ij} =$	e_1	e_2	e_3
$S_i \cup M_{ij}$	$(1,-1,-1,1,-1)$	$e_1^- = 0$	$e_2^{\pm} = 0$	$e_3^{\pm} = 0$
$S_i \cup \hat{M}_{ij}$	$(1,-1,-1,-1,1)$	$e_1^- = 0$	$e_2^{\pm} = 0$	$e_3^{\mp} = 0$
$S_j \cup M_{ij}$	$(-1,1,-1,1,-1)$	$e_1^+ = 0$	$e_2^{\pm} = 0$	$e_3^{\pm} = 0$
$S_j \cup \hat{M}_{ij}$	$(-1,1,-1,-1,1)$	$e_1^+ = 0$	$e_2^{\pm} = 0$	$e_3^{\mp} = 0$
$\hat{\Sigma}_{ij}$	$(-1,-1,1,-1,1)$	$-$	$e_2^{\pm} = 0$	$e_3^{\mp} = 0$

3.4 User defined events

Often dynamical systems are transformed from nonautonomous to autonomous by extending the state variables by adding the (scaled) time as a new variable. Typically, if a systems is periodic, for instance due to a periodic forcing, with period T it might be of interest to locate the end of the period, where time could be reset to zero. Also, a specific time is often used as a Poincaré section when analysing recurrent dynamics. Assume x_k is the state variable of the (scaled) time and t_P is the time that is of interest. Introduce a surface Λ_P defined by the function $h_P(x) = 0$, where $h_P(x) = x_k - t_P$. Then it is only necessary to look for Λ_P for incresing values of $h_P(x)$. Notice that it is possible to introduce any other event function of the user's preference, for example to define other Poincaré surfaces.

3.5 Dificulties using event detection routines in sliding systems

As with any hybrid method, the proposed algorithm can be sensitive to the accuracy of event detection. If an orbit hits a sliding surface $\hat{\Sigma}_{ij}$ almost tangentially, the intersection point will be found with a relatively large error in the sliding direction. Fortunately, this error would then be corrected at the next event, i.e. switching from the sliding to the unconstraned motion at the nearby tangent surface.

Another potentially dangerous phenomenon is *chatter*, i.e., the appearance of infinitely many switches in a finite time interval. Taken literally, the desribed method looks inappropriate for such solutions. However, due to finite time steps, its behaviour in our numerical experiments turns out to be comparable to that of LPC solvers. After successfully computing a (large) number of events, the code merely steps over the rest of them (cf. sec. 4.2).

At points $x \in \Sigma_{ij}$, where both vectors F_i and F_j are tangent to the discontinuity surface, we have

$$\mathcal{L}_{F_j \pm F_i}(h_{ij})(x) = 0$$

and the sliding vector F_{ij} is not defined, see (10). In generic planar Filippov systems without parameters, these *singular sliding points* do not appear. However, they appear in generic one-parameter families of planar Filippov systems as collisions of tangent points (see [29]), as well as in generic n-dimensional parameter-independent Filippov systems with $n \geq 3$ and their families as intersections of the tangent surfaces $\hat{\Sigma}_{ij}^{\pm}$ (see, e.g. [17](Chapter 5)). Therefore, it is also necessary to describe the behaviour of the proposed method near such singularities.

In planar Filippov systems, non-isolated in $\hat{\Sigma}_{ij}$ singular sliding points are not harmfull, since the sliding vector F_{ij} can be extended by continuity to these points (if we neglect infinitely-degenerate cases, see [29]). Note that isolated singular sliding points are usually considered as equilibria. Numerical experiments show that

our method, indeed, steps over non-isolated singular points correctly. In multidimensional Filippov systems, the situation is more involved and requires additional theoretical analysis. However, preliminary numerical experiments demonstrate robust behaviour of the code also in these cases.

3.6 A number of discontinuity surfaces

In the discussion above we have only considered one discontinuity surface. However, it is straightforward to extend the methods to an arbitrary number of surfaces. The only thing one has to do is to assure uniqueness of orbits constrained to the discontinuity surfaces. This is done by only considering a special class of Filippov systems where the vector fields in the different regions of state space are linearly dependent (see further below). To simulate orbits along the various discontinuity surfaces we will here use Filippov's convex method instead of Utkin's equivalence method (as was the case in sec. 2) to show how that approach can be implemented but also since it is slightly more straight forward. Notice also that the notation in this section therfore differs somewhat from the one-surface case.

Assume we have a general dynamical system (1) and M discontinuity surfaces Σ_i defined by M functions $h_i(x) = 0$. These surfaces divide the state space into a number of disjoint sets, where the vector fields are different. As mentioned above we have specific rules on how the vectorfields can look like to quarantee uniqueness of orbits constrained to the disconinuity surfaces. The idea we propose here is to introduce a base vector field that is valid when $h_i(x) > 0$ for all i and for each surface crossing we make an addition Δ_i to the base vector field that is used after the surface crossing. Using this idea and Filippov's convex method the vector field for the whole domain, including the sliding regions, can be written as

$$F(x) = F_{\text{orig}}(x) + \sum_{i=1}^{M} \Delta_i(x)\mu_i(x), \tag{23}$$

where

$$\mu_i(x) = \begin{cases} 1, & h_i(x) > 0, \\ [0,1] & h_i(x) = 0, \\ 0, & h_i(x) < 0. \end{cases} \tag{24}$$

For each surface we have thus defined a function $\mu_i(x)$, in a similar way as in the single surface case, that determines the active vector field (cf. eqs. (9) and (12)).

In the same way as described earlier in secs. 3.2-3.5 the program has to keep track of if surfaces have been crossed or not so that the the correct values of the μ_is are used, and thus the correct vector field is integrated. The only difference from the one-surface case is that each discontinuity surface requires its own set of event variables and event functions.

Since sliding can occur along more the one discontinuity surface simultaneously the values of the corresponding $\mu_i \in [0,1]$ have to be determined. This can be done by using the same approach as in the one-surface case. To give an example, and without lack of generality, assume that the orbit will slide along the surfaces Σ_i and Σ_j then we know that $h_i(x) = 0$ and $h_j(x) = 0$, respectively, and we want $\mathcal{L}_F(h_i)(x) = 0$ and $\mathcal{L}_F(h_j)(x) = 0$ to hold. By using these conditions, eqs. (23) and (24) we get that

$$\begin{pmatrix} \mu_i(x) \\ \mu_j(x) \end{pmatrix} = -\begin{pmatrix} \mathcal{L}_{\Delta_i} h_i(x) & \mathcal{L}_{\Delta_j} h_i(x) \\ \mathcal{L}_{\Delta_i} h_j(x) & \mathcal{L}_{\Delta_j} h_j(x) \end{pmatrix}^{-1} \begin{pmatrix} \mathcal{L}_{\tilde{F}}(h_i)(x) \\ \mathcal{L}_{\tilde{F}}(h_j)(x) \end{pmatrix}, \tag{25}$$

where

$$\tilde{F}(x) = F_{\text{orig}}(x) + \sum_{k=1,\ k\neq i,j}^{M} \Delta_k(x)\mu_k(x). \tag{26}$$

Furthermore, a similar regularization of the sliding vector field as in sec. 3.1 (see especially eq. 13) is possible by introducing M constants C_i, where each such constant has a positive value if the systems is sliding along the ith surface and is zero otherwise.

The main disadvantage with this approach is that the number of surfaces and event locations grows quickly with the number of discontinuity surfaces which naturally increses the simulation time. However, this approach for two surfaces has been implemented in the drill-string example in sec. 4.3.

4 Examples

In this section we will present some results of using the implemented programs described in this paper, and presented in appendix A, to simulate three different systems. These systems have been chosen to highlight the variety of Filippov systems and different dynamical behaviour that the simulation tool can handle. The first system is a nonlinear dry-friction oscillator with one discontinuity surface that has a rich dynamics and chartacteristic *stick-slip* motion. The second system is a linear relay feedback system with one discontinuity surface that exhibit chaotic behaviour. The third example is a nonlinear drill-string system with two discontinuity surfaces. For these examples we will give all necessary information so that the MATLAB simulation routines in appendix A can be used. In all examples the ODE-solver `ode45` (4th order Runge-Kutta) has been used with MATLAB's built-in event detection routines. Also, all files used in these examples can be downloaded from `http://seis.bris.ac.uk/~enptp/Filippov/`.

4.1 A dry-friction oscillator

Sick-slip motion is a well-known, but not fully understood, behaviour in many mechanical systems with friction. The most simple examples showing this kind of behaviour are dry-friction oscillators. Therefore they have drawn a lot of attention and been widely studied. Here we will focus on an undamped dry-friction oscillator with one degree of freedom given by

$$\ddot{y} + y = \sin(\omega t) - F\text{sign}(\dot{y}), \tag{27}$$

where ω is the forcing frequency and F is the size of the Coloumb friction force. This system has been described by [16] and extensively analysed in [27, 10, 34].

To be able to use the proposed strategy for integrating (27) we will consider the equivalent autonomous first order system

$$\dot{x} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} x_2 \\ -x_1 + \sin(x_3) - F\text{sign}(x_2) \\ \omega \end{pmatrix}, \tag{28}$$

where $x = (x_1, x_2, x_3)^T = (y, \dot{y}, \omega t \mod 2\pi)^T$. This implies that the discontinuity surface Σ_{12} is defined as

$$\Sigma_{12} = \left\{ x \in \mathbb{R}^3 \mid H_{12}(x) = 0 \right\},$$

which divides the state space into two disjoint regions

$$S_1 = \left\{ x \in \mathbb{R}^3 \mid H_{12}(x) > 0 \right\}, \quad S_2 = \left\{ x \in \mathbb{R}^3 \mid H_{12}(x) < 0 \right\},$$

where

$$H_{12}(x) = x_2.$$

Now it is easy to rewrite (28) as

$$\dot{x} = \begin{cases} F_1(x), & x \in S_1 \\ F_2(x), & x \in S_2, \end{cases} \tag{29}$$

where

$$F_1 = \begin{pmatrix} x_2 \\ -x_1 + \sin(x_3) - F \\ \omega \end{pmatrix}, \quad F_2 = \begin{pmatrix} x_2 \\ -x_1 + \sin(x_3) + F \\ \omega \end{pmatrix}.$$

The first approach to analyse the dynamics of a system of this kind is often to make a parameter sweep and create a brute-force bifurcation diagram, which could reveal stable attractors for the given parameter ranges. In Fig. 4(a) a brute-force bifurcation diagram of the system (29) is depicted, where the parameter F was kept fixed at 0.4 and the frequency ω varied. For each value of the frequency the system was integrated 500 forcing periods (corresponding to an integration time $500 \times 2\pi/\omega$) and for the final 100 periods the variable x_2 was recorded and plotted every time the orbit reached the Poincaré section defined by the function

$$h_P(x) = x_3 - 2\pi.$$

It is clear from Fig. 4(a) that we have a big peak at $\omega = 1$ due to resonance, as expected. Further we see that there are drastic changes in the bifurcation diagram at $1/\omega \approx 1.8$ and $1/\omega \approx 4.8$. To understand how the dynamics differ from one parameter range to the next the limit cycles corresponding to the frequencies at the Roman numerals 'I', 'II' and 'III' in Fig. 4(a) was located and are shown in Fig. 4(b), (c) and (d), respectively. In Fig. 4(b), where $1/\omega = 0.5$, we see a limit cycle without any sliding segments, but as we increase $1/\omega$ to 3 (point 'II' in Fig. 4(a)) we clearly see in Fig. 4(c) that the limit cycle has two segments of sliding motion. Finally, as we increase $1/\omega$ further to 6 (point 'III' in Fig. 4(a)) the limit cycle include an even greater number of sliding segments, as seen in Fig. 4(d). These results indicate that at least two nonsmooth transitions (or sliding bifurcations [29, 13], see sec. 1) has occured along the parameter sweep. The proposed sumulation algorithm can now be extended to accurately detect where these transitions occur. In fact, in [10] the transitions points of this particular system are accurately detected and the sliding bifurcations are continued under parameter variations, and a two-parameter bifurcation diagram (in F and ω) is presented to show how the sliding bifuractions organize the global dynamics. The continuation methods used in [10] are presented in great detail in [34].

This particular example shows that the proposed algorithm cannot only be used for direct numerical simulations of systems with sliding motion but also as a building block for continuation algorithms that follow branches of sliding bifurcations. The results have helped us to understand what happens to the dynamics in mechanical systems with sliding segments (also referred to as *stick-slip* motion).

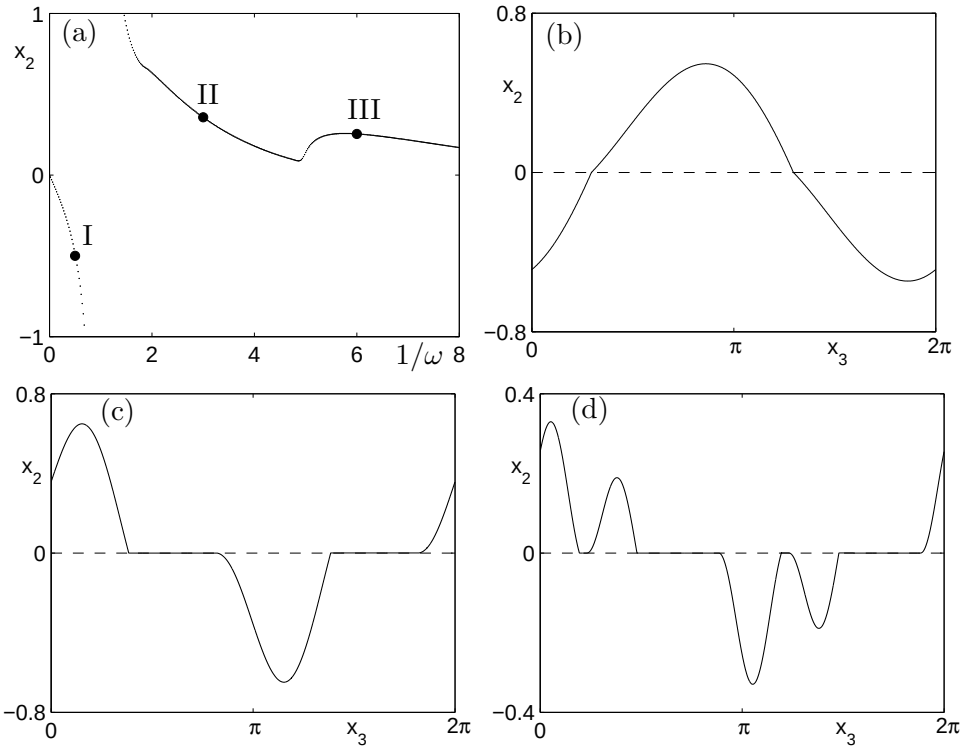

Figure 4: (a) A brute-force bifurcation diagram for recurrent motion, where x_2 is plotted against $1/\omega$ at the Poincaré section $x_3 = 2\pi$. The parameter values at the indicated points 'I', 'II' and 'III' are $(F = 0.4, \omega = 0.5)$, $(F = 0.4, \omega = 3)$ and $(F = 0.4, \omega = 6)$, respectively. The panels (b), (c) and (d) show the limit cycles corresponding respectively to the points 'I', 'II' and 'III'.

4.2 A relay feedback system

Relay feedback is one of the most commonly used control techniques in practical applications, such as temperature control and mechanical and electro-mechanical systems. A single-input single-output relay feedback system can be written as

$$\dot{x} = Ax + Bu, \tag{30}$$
$$y = Cx, \tag{31}$$
$$u = -\mathrm{sgn}(y), \tag{32}$$

or

$$\dot{x} = \begin{cases} Ax - B, & Cx > 0 \\ Ax + B, & Cx < 0 \end{cases} \tag{33}$$

where $x \in \mathbb{R}^n$ is the state vector, and $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times 1}$, $C \in \mathbb{R}^{1\times n}$ are constant matrices. Here we will take a closer look at a particular system analysed in [12], where

$$A = \begin{pmatrix} -(2\zeta\omega + 1) & 1 & 0 \\ -(2\zeta\omega + \omega^2) & 0 & 1 \\ -\omega^2 & 0 & 0 \end{pmatrix}, B = \begin{pmatrix} 1 \\ -2\sigma \\ 1 \end{pmatrix} \text{ and } C = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \tag{34}$$

and where $x = (x_1, x_2, x_3)^T$ is the state vector and $k \in \mathbb{R}$ is the control parameter. From this we can also conclude that the discontinuty surface Σ_{12} is defined by

$$\Sigma_{12} = \left\{ x \in \mathbb{R}^3 \mid H_{12}(x) = 0 \right\}, \tag{35}$$

and the two disjoint regions are thus given by

$$S_1 = \left\{ x \in \mathbb{R}^3 \mid H_{12}(x) > 0 \right\} \text{ and } S_2 = \left\{ x \in \mathbb{R}^3 \mid H_{12}(x) < 0 \right\}, \tag{36}$$

where

$$H_{12}(x) = x_1. \tag{37}$$

To be able to use our method we write the dynamical system (33) as

$$\dot{x} = \begin{cases} F_1(x), & x \in S_1, \\ F_2(x), & x \in S_2, \end{cases} \tag{38}$$

where the two vector fields are

$$F_1(x) = \begin{pmatrix} -(2\zeta\omega + 1)x_1 + x_2 - 1 \\ -(2\zeta\omega + \omega^2)x_1 + x_3 + 2 \\ -\omega^2 x_1 + x_2 - 1 \end{pmatrix}, \; F_2(x) = \begin{pmatrix} -(2\zeta\omega + 1)x_1 + x_2 + 1 \\ -(2\zeta\omega + \omega^2)x_1 + x_3 - 2 \\ -\omega^2 x_1 + x_2 + 1 \end{pmatrix}. \tag{39}$$

In Fig. 5(a) we see the dynamics, and eventually the limit cycle, for a particular

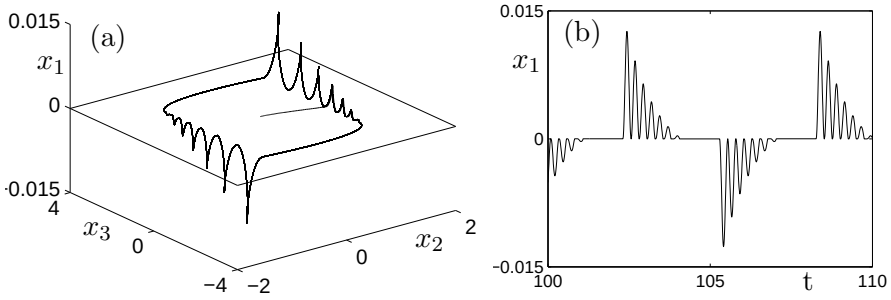

Figure 5: (a) A state space diagram and (b) a time-history diagram of the state variable x_1 showing the eventual stable periodic motion, for $\zeta = 0.05$ and $\omega = 25$.

parameter combination ($\zeta = 0.05$, $\omega = 25$) showing both unconstrained and sliding motions. The initial condition is on the sliding surface, as seen in Fig. 5. This does not represent a problem for the simulator. Note also the large number of events that take place per period including 18 separate departures from the sliding surface.

A brute force bifurcation diagram, where $\zeta = -0.07$ was held fixed and σ varied, is shown in Fig. 6(a). There we see a period-adding sequence where high periodic motion or chaos is observed between periodic windows, and where each period include many intervals of sliding motion. Similar period-adding behaviour can also be seen in impacting systems and piecewise linear maps. In Fig. 6(b) chaotic motion is depicted for $\zeta = -0.07$ and $\sigma = 10$ to highlight the great complexity of the motion.

Here we have shown that the simulation method can be used to analyse systems with a high number of sliding segments per period by calculating bifurcation diagrams. Together with a continuation code, as discussed in sec. 4.1, a bifurcation diagram constitute a powerful tool to analyse linear Filippov system that arise in many electronical systems.

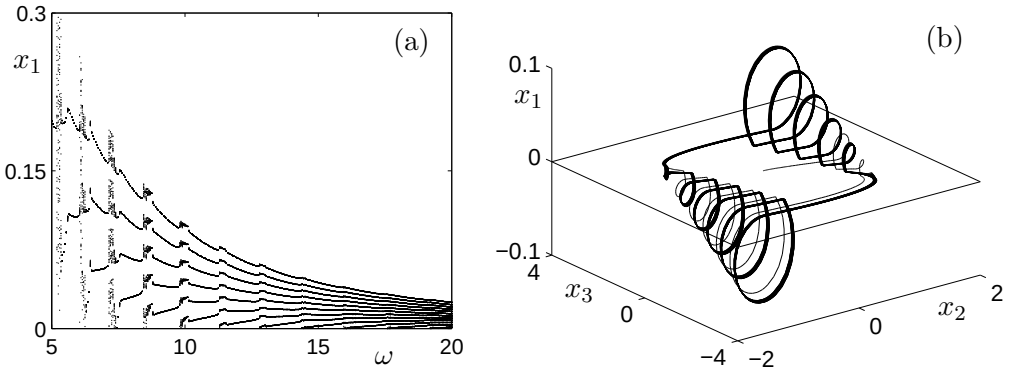

Figure 6: (a) A bifurcation diagram showing x_1 versus ω for $\zeta = -0.07$, where the Poincaré surface is given by $\mathcal{L}_{F_1}(H)(x) = 0$. (b) A state space diagram for $\zeta = -0.07$, $\omega = 10$), where the initial condition is taken to be on the sliding surface.

4.3 A drill-string system

A simple model of the motion of a drill-string consists of two discs separated by a string with a brake connected to the lower disc. The upper disc represents a rotary table to which a drive motor is connected via a gear box and the string. The lower disc represents the drill-string with the bottom-hole-assembly at the drill-rig and the additional break implements the friction forces between the drill bit and the bore whole. Such a drill-string system is described in [31] and modelled by

$$J_u\ddot{\theta}_u + k_\theta\left(\theta_u - \theta_l\right) + T_{fu}\left(\theta_u\right) = k_m u, \tag{40}$$

$$J_l\ddot{\theta}_l - k_\theta\left(\theta_u - \theta_l\right) + T_{fl}\left(\theta_l\right) = 0, \tag{41}$$

where θ_u (θ_l), J_u (J_l) and T_{fu} (T_{fl}) are respectively the angle, the moment of inertia and friction torque of the upper (lower) disc. Furthermore, k_θ is the torsional stiffness of the drill string, u is the input voltage and k_m is a motor constant. Making the assumption used in [31] on the friction laws at the upper an lower discs we have

$$T_{fu}(\dot{\theta}_u) = \begin{cases} T_{fu}^+ = T_{sup} + b_{up}\dot{\theta}_u, & \dot{\theta}_u > 0 \\ T_{fu}^- = -T_{sun} + b_{un}\dot{\theta}_u, & \dot{\theta}_u < 0 \end{cases} \tag{42}$$

and

$$T_{fl}(\dot{\theta}_l) = \begin{cases} T_{fl}^+ = T_{sl} + T_1\left(1 + \dfrac{2}{1+e^{\beta_1\dot{\theta}_l}}\right) + T_2\left(1 + \dfrac{2}{1+e^{\beta_2\dot{\theta}_l}}\right), & \dot{\theta}_l > 0, \\ T_{fl}^- = -T_{sl} - T_1\left(1 + \dfrac{2}{1+e^{-\beta_1\dot{\theta}_l}}\right) - T_2\left(1 + \dfrac{2}{1+e^{-\beta_2\dot{\theta}_l}}\right), & \dot{\theta}_l < 0, \end{cases} \tag{43}$$

where b_{up}, b_{un}, T_{sup}, T_{sun}, T_{sl}, T_1, T_2, β_1 and β_2 are all constants. By letting

$$x = (x_1, x_2, x_3)^T = (\theta_u - \theta_l, \dot{\theta}_u, \dot{\theta}_l)^T$$

the drill-string system (40-(41) can be written as

$$\dot{x} = \begin{pmatrix} x_2 - x_3 \\ \dfrac{k_m}{J_u}u - \dfrac{k_\theta}{J_u}x_1 - \dfrac{1}{J_u}T_{fu}(x_2) \\ k_\theta x_1 - \dfrac{1}{J_l}T_{fl}(x_3) \end{pmatrix}. \tag{44}$$

From (42) and (43) we can determine that there are two discontinuity surfaces Σ_1 and Σ_2 defined by

$$\Sigma_1 = \left\{ x \in \mathbb{R}^3 \mid H_1(x) = 0 \right\}, \quad \Sigma_2 = \left\{ x \in \mathbb{R}^3 \mid H_2(x) = 0 \right\}, \tag{45}$$

where

$$H_1(x) = x_2 \quad \text{and} \quad H_2(x) = x_3. \tag{46}$$

Notice that we are using the methodology and notation introduced in sec. 3.6. Now it is straight forward to write (44) in our preferred format:

$$\dot{x} = F_1(x) + \Delta_1(x)\mu_1(x) + \Delta_2(x)\mu_2(x), \tag{47}$$

where

$$F_1(x) = \begin{pmatrix} x_1 - x_2 \\ \dfrac{k_m}{J_u}u - \dfrac{k_\theta}{J_u}x_1 - T_{fu}^+(x_2) \\ \dfrac{k_\theta}{J_l}x_1 - T_{fl}^+(x_3) \end{pmatrix}, \tag{48}$$

$$\Delta_1(x) = \begin{pmatrix} 0 \\ T_{fu}^+(x_2) - T_{fu}^-(x_2) \\ 0 \end{pmatrix} \text{ and } \Delta_2(x) = \begin{pmatrix} 0 \\ 0 \\ T_{fl}^+(x_2) - T_{fl}^-(x_2) \end{pmatrix}. \tag{49}$$

Here it is important to point out that the four possible vector fields are linearly dependent, so that we have a uniquely defined sliding vector field in the intersection between discontinuity surfaces. Since this system has two dicontinuity surfaces, due to the Coloumb friction models at the upper and lower discs, it it possible for the discontinuity surfaces to cross, which practically means that the upper and lower disc can get stuck at the same time. However, for the example examined here

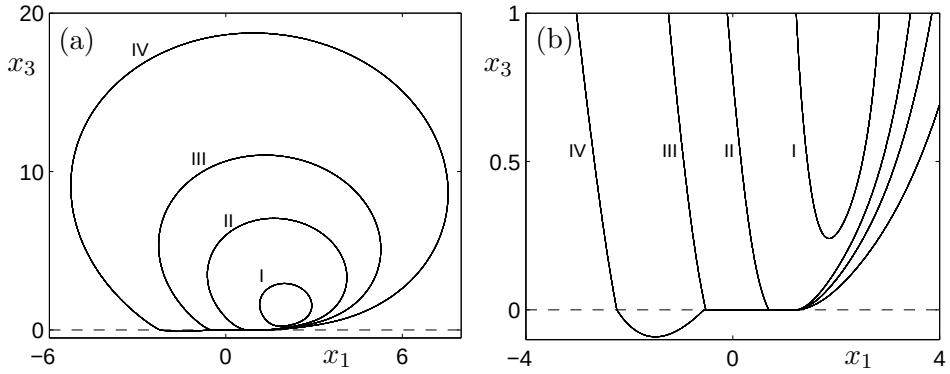

Figure 7: (a) A projection of the state diagram showing x_3 against x_1 of four limit cycles 'I', 'II', 'III' and 'IV' corresonding to $u = 1$, $u = 2$, $u = 3$ and $u = 5$, respectively. (b) A close up of panel (a) to show the transformation from the non-sliding limit cycle to sliding limit cycles via two sliding bifurcations. The rest of the system parameters are $k_m = 3.5693$, $k_\theta = 0.078$, $J_u = 0.4765$, $J_l = 0.0326$, $T_{sup} = 0.3216$, $T_{sun} = 0.3026$, $b_{up} = b_{un} = 1.9667$, $T_{sl} = 0.0940$, $T_1 = 0.0826$, $T_2 = -0.2910$, $\beta_1 = 6.3598$ and $\beta_2 = 0.0786$

limit cycles with sliding along one surface (Σ_2) only has been found. In Fig. 7 we

show four limit cycles 'I', 'II', 'III' and 'IV' corrsponding to output voltages $u = 1$, $u = 3$, $u = 3$ and $u = 5$, respectively. The limit cycle 'I' does not have any sliding segments but as the voltage is increased to 2 the limit cycle has undergone a grazing-sliding bifurcation and both the periodic solutions 'II' and 'III' clearly have a sliding segment. As the voltage is increased further the limit cycle changes its character via a *switching-sliding* bifurcation. As seen in Fig. 7(b) the limit cycle has an extra discontinuity surface crossing, but the number of sliding segments remain the same.

This is a good example where numerical experiments show how dynamics change under parameter variations, thus giving an impulse for further analytical and numerical investigation.

5 Conclusions and outlook

The introduced methods give a simple way to automatically simulate generic orbits of Filippov systems using a hybrid system approach, and where the user only has to introduce information about the vector fields and discontinuity surfaces. Although many reserch groups have developed simulation environments for specific nonsmooth problems, the authors are unaware of a general Filippov system solver, as the one proposed here. Therefore we hope that this software, and the ideas upon which it is built, will be used by both applied mathematicians and engineers for such simulations, and hopefully make the whole community to think of these problems in a more general setting.

The obvious strength of the proposed simulation code is that it is relatively simple to use, as long as the user has some knowledge of MATLAB. One problem with the hybrid system approach though is that the combinatotic complexity of the code grows quickly as the number of discontinuity surfaces are incread. Also, we choose MATLAB since it is relatively simple to test ideas, implement code and use the final product. The down side is of course that MATLAB is much slower than an equivalent program written in, for instance, `C`, `C++` or FORTRAN. So if the code is to be used for larger scale problems, e.g. with many discontinuity surfaces, such an implementation is prefereable. Also, for even more efficient calcultaions it would useful to have a script that intially generates a code for the user-specific problem, in terms of the state space dimension and the number of discontinuity surfaces.

As mentioned earlier, the algorithm can also be used as a building block for a continuation algorithm that follow both periodic orbits in one parameter and codimension-one bifurcations in two parameters, using Poincaré maps (i.e. shooting). Such an implementation will be discussed in future works [34, 10]. We also hope to develop a general simulation and continuation interactive environment that supports both discontinuous vector fields and state jumps, so that the user specifies a surface as continuous, Filippov or impacting.

Acknowledgment

This work was supported by the EU FP5 Project SICONOS (Grant no. IST-2001-37172). The authors would like to thank Alan Champneys and Arne Nordmark for valuable comments.

References

[1] V. I. Arnol'd. *Ordinary Differential Equations*. MIT Press, Cambridge, MA, 1973.

[2] U. M. Ascher and L. R. Petzold. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. SIAM, Philadelphia, USA, 1998.

[3] R. Ashino, M. Nagase, and R. Vaillancourt. Behind and beyond the MATLAB ODE suite. *Comput. Math. Appl.*, 40:491–512, 2000.

[4] B. Brogliato. Some perspectives on the analysis and control of complementarity systems. *IEEE Transactions on Automatic Control*, 48:918–935, 2003.

[5] P. N. Brown, G. D. Byrne, and A. C. Hindmarsh. VODE: a variable-coefficient ODE solver. *SIAM J. Sci. Statist. Comput.*, 10:1038–1051, 1989.

[6] J. C. Butcher, editor. *Special issue celebrating the centenary of Runge-Kutta methods*. North-Holland Publishing Co., Amsterdam, 1996. Appl. Numer. Math. **22** (1996), no. 1-3.

[7] E. Celledoni. Discrete QMR and BGC in the numerical solution of linear systems of ODEs. *J. Comput. Appl. Math.*, 91:159–177, 1998.

[8] J. J. B. de Swart. A simple ODE solver based on 2-stage Radau IIA. *J. Comput. Appl. Math.*, 84:277–280, 1997.

[9] F. Dercole and Yu. A. Kuznetsov. *SlideCont: An Auto97 driver for sliding bifurcation analysis*. Department of Mathematics, Universiteit Utrecht, The Netherlands, 2002.

[10] M. di Bernardo, S. J. Hogan, P. Kowalczyk, and P. T. Piiroinen. Numerical detection and continuation of sliding bifurcations in a dry-friction oscillator. In preparation., 2005.

[11] M. di Bernardo, K. H. Johansson, and F. Vasca. Sliding orbits and their bifurcations in relay feedback systems. In *Proc. 38th IEEE Conference on Decision and Control*, Phoenix, AZ, 1999.

[12] M. di Bernardo, K. H. Johansson, and F. Vasca. Self-oscillations and sliding in relay feedback systems: Symmetry and bifurcations. *International Journal of Bifurcations and Chaos*, 11(4):1121–1140, 2001.

[13] M. di Bernardo, P. Kowalczyk, and A. Nordmark. Bifurcations of dynamical systems with sliding: derivation of normal-form mappings. *Physica D*, 170:175–205, 2002.

[14] E. J. Doedel, A. R. Champneys, T. F. Fairgrieve, Yu. A. Kuznetsov, B. Sandstede, and X.-J. Wang. *Auto97: Continuation and bifurcation software for ordinary differential equaitions (with HomCont)*. Computer Science, Concordia University, Montreal, Canada, `ftp.cs.concordia.ca/doedel/doc/auto`, 1997.

[15] W. Enright. On the efficient and reliable numerical solution of large linear systems of ODE's. *IEEE Trans. Automat. Control*, 24:905–908, 1979.

[16] M. I. Feigin. *Forced Oscillations in Systems With Discontinuous Nonlinearities*. Nauka, Moscow, 1994. In Russian.

[17] A. F. Filippov. *Differential Equations with Discontinuous Righthand Sides*. Kluwer Academic Publishers, Dortrecht, 1988.

[18] U. Galvanetto. Some discontinuous bifurcations in a two block stick-slip system. *Journal of Sound and Vibration*, 284(4):653 – 669, 2001.

[19] U. Galvanetto and S. R. Bishop. Dynamics of a simple damped oscillator undergoing stick-slip vibrations. *Meccanica*, 34:337–347, 2000.

[20] C. W. Gear and O. Østerby. Solving ordinary differential equations with discontinuities. *ACM Trans. Math. Software*, 10:23–44, 1984.

[21] J. Guckenheimer and P. Holmes. *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields*. Springer–Verlag, New York, 1983. Applied Mathematical Sciences, Volume 42.

[22] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving ordinary differential equations. I Nonstiff problems*, volume 8 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, second edition, 1993.

[23] E. Hairer and G. Wanner. *Solving ordinary differential equations. II Stiff and differential-algebraic problems*, volume 14 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, second edition, 1996.

[24] E. Hairer and G. Wanner. Stiff differential equations solved by Radau methods. *J. Comput. Appl. Math.*, 111:93–111, 1999.

[25] A. C. Hindmarsh. ODEPACK, a systematized collection of ODE solvers. In *Scientific computing (Montreal, Que., 1982)*, IMACS Trans. Sci. Comput., I, pages 55–64. IMACS, New Brunswick, NJ, 1983.

[26] M. Jean. The non-smooth contact dynamics method. *Computer Methods in Applied Mechanics and Engineering*, 177(3-4):235–257, 1999.

[27] P. Kowalczyk. *Analytical and Numerical Investigations of Sliding Bifurcations in n-dimensional piecewise- smooth systems*. PhD thesis, Department of Engineering Mathematcics, University of Bristol, Bristol, UK, 2003.

[28] Yu. A. Kuznetsov. *Elements of Applied Bifurcation Theory*. Springer–Verlag, New York, 3rd edition, 2004. Applied Mathematical Sciences, Volume 112.

[29] Yu. A. Kuznetsov, S. Rinaldi, and A. Gragnani. One-parameter bifurcations in planar Fillipov systems. *Internat. J. Bifur. Chaos Appl. Sci. Engrg.*, 13:2157–2188, 2003.

[30] R. I. Leine. *Bifurcations in Discontinuous Mechanical Systems of Filippov-Type*. PhD thesis, Teknische Universiteit Eindhoven, The Netherlands, 2000.

[31] N. Miajlovic, A. A. van Veggel, N. van de Wouw, and H. Nijmeijer. Analysis of friction-induced limit cycling in an experimental drill-string system, 2004. Preprint.

[32] J. J. Moreau. Numerical aspects of the sweeping process. *Computer Methods in Applied Mechanics and Engineering*, 177(3-4):329–349, 1999.

[33] B. V. Pavlov and O. E. Rodionova. Numerical solution of systems of linear ordinary differential equations with constant coefficients. *Comput. Math. Math. Phys.*, 34:535–539, 1994.

[34] P. T. Piiroinen. Numerical detection and continuation of sliding bifurcations in Filippov systems using an event-driven simulator, 2005. In preparation.

[35] L. F. Shampine. *Numerical solution of ordinary differential equations*. Chapman & Hall, New York, 1994.

[36] L. F. Shampine and M. W. Reichelt. The MATLAB ODE suite. *SIAM J. Sci. Comput.*, 18:1–22, 1997.

[37] V. I. Utkin. *Sliding Modes in Control Optimization*. Springer–Verlag, New York, 1992.

A How to use the code

The method described in the present paper in the case of a single discontinuity surface has been implemented in MATLAB. Here we will give some details on how to use the the programs. Templates for the files can be found at `http://seis.bris.ac.uk/~enptp/Filippov/`. There one can also find the files used for the examples in sec. 4, including the drill-string example with two discontinuity surfaces.

The program consists of three files,

```
run_oscillator.m
filippov.m
vectorfields.m
```

and possibly the optional files

```
pfunction.m
jacobians.m
```

The file `run_oscillator.m` is the main program in which integration time, parameters, initial conditions, names of the vector field, Jacobian matrix, and Poincaré function files, the name of the ODE solver and its properties are introduced by the user. This file also calls the file `filippov.m`, which takes care of the event handling and also calls the MATLAB built-in ODE solver and event detector. In the vector field file `vectorfields.m`, which is used by `filippov.m`, the user adds the different vector fields, the function defining the discontinuity surface, the normal to the discontinuity surface, and possibly the function defining a Poincaré surface. In `pfunction.m` the user defines the action to be taken at the Poincaré surface. Finally, to make grazing location in the sliding surface more robust the Jacobian matrices corresponding to the different vector fields can be added.

It has to be noted that no particular effort has been done to optimize the code at this stage. The main reason is to make it as transparent as possible so that users can do suitable changes themselves.

Let us now show explicitly what to write in the different files specified by the user. The system we will use in this exposition is the same dry-friction oscillator as described in sec. 4.1, where a more thorough introduction to this system is done. Let the dynamical system be given by

$$\dot{x} = \begin{cases} F_1(x), & x_2 > 0, \\ F_2(x), & x_2 < 0, \end{cases} \tag{50}$$

where $x = (x_1, x_2, x_3)^T$,

$$F_1(x) = \begin{pmatrix} x_2 \\ -x_1 + \sin(x_3) - F \\ \omega \end{pmatrix}, \quad F_2(x) = \begin{pmatrix} x_2 \\ -x_1 + \sin(x_3) + F \\ \omega \end{pmatrix}, \tag{51}$$

and F and ω are constants. The user needs to adjust the different files as follows.

run_oscillator.m

The following has to be given by the user:

ODE-solver

`solver = 'ode45';` (The choice of MATLAB ODE solver)

MATLAB's ODE-solver properties

```
opts = odeset('RelTol',1e-6,'AbsTol',1e-6,'MaxStep',0.1);
```

Name of the vector field file

```
vfields = 'vectorfields';
```

Name of the Jacobians file

```
vfields = 'jacobians';
```

Name of the Poincaré function file

```
pfunction = 'pfunction';
```

Filippov parameter

```
C = 1;
```

Parameter list

```
F = 0.4; omega = 3; params = [F,omega];
```

Initial conditions

```
y0 = [1,2,0];
```

Integration time

```
T = 2*pi/omega; tspan = [0,T];
```

The output that will be given by `filippov.m` is a list of the time `t`, and the corresponding values of the states `y`, the times `te` at the events, the state vector `ye` at the events, an index list `ie` of the events, and the event variables `se` at the events (cf. the event handling in MATLAB).

vectorfields.m

In this file the user has to specify the following:

Parameters

```
F = params(1); omega = params(2);
```

The two vector fields

```
F1 = [y(2);-y(1)-sin(y(3))-F;omega];
```
($H > 0$)

```
F2 = [y(2);-y(1)-sin(y(3))+F;omega];
```
($H < 0$)

The function defining the discontinuity surface

```
H = y(2);
```

The gradient of H

```
dH = [0,1,0];
```

The Poincare surface

```
h = y(3)-2*pi;
```
(The system is 2π-periodic in x_3.)

Location direction for the Poincare section

```
dir = 1;
```

jacobians.m

In this file the user has to specify the following:

Parameters

(No parameters needed)

The two Jacobians

```
J1 = [0,1,0; -1,0,-cos(y(3)); 0,0,0];
```
($H > 0$)

```
J2 = J1;
```
($H < 0$)

The second derivative of H

```
d2H = zeros(3,3);
```

pfunction.m

In this file the user has to specify the following:

Parameters

(No parameters needed)

Action at Poincaré section

`y1 = [y(1),y(2),0];` (Reset of the scaled time.)